HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 100201439-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	Henry Chang et al.	Confirmation No.:	7781
Application No.:	10/681,556	Examiner:	Ben C. Wang
Filing Date:	October 8, 2003	Group Art Unit:	2192

Title: SOFTWARE TESTING

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on Feb. 11, 2008.

☒ The fee for filing this Appeal Brief is $510.00 (37 CFR 41.20).

□ No Additional Fee Required.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

□ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

□ 1st Month $120 □ 2nd Month $460 □ 3rd Month $1050 □ 4th Month $1640

□ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of $ 510 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

☒ A duplicate copy of this transmittal letter is enclosed.

□ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450

Date of Deposit:

OR

☒ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.

Date of facsimile: April 10, 2008

Typed Name: Judy H. Chung

Signature:

Respectfully submitted,

Henry Chang et al.

By

Tiep H. Nguyen

Attorney/Agent for Applicant(s)

Reg No. : 44,465

Date : April 10, 2008

Telephone : (703) 652-3819

Rev 10/07(AplBrief)

Total number of pages: 24

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Attorney Docket No.: 100201439-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	Henry Chang et al.	**Confirmation No.:**	7781
Serial No.:	10/681,556	**Examiner:**	Ben C. Wang
Filed:	October 8, 2003	**Group Art Unit:**	2192
Title:	SOFTWARE TESTING		

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF - PATENTS

Sir:

This is an Appeal Brief in connection with the decisions of the Examiner in an Office Action dated November 14, 2007. This application has been more than twice rejected. Each of the topics required in an Appeal Brief and a Table of Contents are presented herewith and labeled appropriately.

(1) Real Party In Interest

The real party in interest is Hewlett-Packard Development Company, L.P.

(2) Related Appeals And Interferences

There are no other appeals or interferences related to this case.

(3) Status Of Claims

Claim 3 has been canceled. Claims 1, 2, and 4-17 are pending and rejected. All pending claims 1, 2, and 4-17 are hereby appealed.

(4) Status of Amendments

No amendment was filed subsequent to the last Office Action dated November 14, 2007. There was no final rejection.

(5) Summary Of Claimed Subject Matter

It should be understood that the claimed subject matter is supported in at least the following cited sections of the present application. Thus, other sections in the present application may provide the same or additional supports as well.

In claim 1, a method for testing software, comprising:

examining an application software program including calls to system classes with both a static analysis tool and a dynamic analysis tool (page 17, lines 17+);

determining a static use count of said system classes from the examining (page 17, lines 17+);

deriving a dynamic use count of each of said system classes during operation of said application software program from the examining (page 17, lines 17+);

assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count (page 17, lines 17+); and

testing said system classes in order according to said corresponding proportional weighing attributes (page 17, lines 17+).

In claim 10, a machine-readable medium on which is encoded machine-readable code for testing object-oriented system software having system classes, the machine readable code comprising:

machine-readable code for running a static analysis tool for examining an application software program, the application software program including calls to the system classes (page 2, lines 31+; FIG. 1 at 104);

machine-readable code for determining a static use count of the system classes in the application software program from the result (page 2, lines 31+; FIG. 1 at 104);

machine-readable code for running a dynamic analysis tool for examining the application software program and producing a dynamic use count based on the application software program's dynamic use of the system functions while running the application software program (page 2, lines 31+; FIG. 1 at 104);

machine-readable code for assigning to each system class a weight based on the static use count and the dynamic use count (page 2, lines 31+; page 8, lines 9-31; FIG. 1 at 102 and 104), and

machine-readable code for testing the system classes, in order, based on the assigned weight, from a first entry having a greatest weight (page 10, lines 11+; FIG. 1 at 112).

In claim 15, a machine-readable medium on which is encoded a software tester program code, the software tester program code comprising:

means for examining an application software program including calls to system classes with both a static analysis tool and a dynamic analysis tool (page 2, lines 28+; FIG. 1 at 100);

means for determining a static use count of said system classes from the examining (page 2, lines 28+; FIG. 1 at 102);

means for deriving a dynamic use count of each of said system classes during operation of said application software program from the examining (page 2, lines 28+; FIG. 1 at 106);

means for assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count (page 2, lines 31+; page 8, lines 9-31; FIG. 1 at 102 and 104); and

means for testing said system classes in order according to said corresponding proportional weighing attributes (page 10, lines 11+; FIG. 1 at 112).

In claim 17, a business model for testing software, comprising:

setting a resource limit on the available time or money that is devoted to testing a particular application software program (page 17, lines 17+);

examining said application software program including calls to system classes with both a static analysis tool and a dynamic analysis tool (page 17, lines 17+);

determining a static use count of said system classes from the examining (page 17, lines 17+);

deriving a dynamic use count of each of said system classes during operation of said application software program from the examining (page 17, lines 17+);

assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count (page 17, lines 17+);

testing said system classes in order according to said corresponding proportional weighing attributes and proceeding down to the least heavily weighted system classes (page 17, lines 17+); and

stopping testing when said resource limit is reached (page 17, lines 17+).

(6) Grounds of Rejection to be Reviewed on Appeal

a) Whether claims 1, 4, 5, 8-12, 15, and 17 were properly rejected under 35 U.S.C. §103(a) as being unpatentable over Ur et al. (US 2003/0110474) in view of Cahill et al. ("The Java Metrics Reporter – An Extensible Tool for OO Software Analysis," 2002 IEEE), Benlarbi et al. ("Polymorphism Measures for Early Risk Prediction," 1999, ACM), and Mitchell et al. ("Towards a definition of run-time object-oriented metrics," July 22, 2003, ECOOP).

b) Whether claims 6 and 13 were properly rejected under 35 U.S.C. §103(a) as being unpatentable over Ur et al. in view of Cahill et al., Benlarbi et al., Mitchell et al., and further in view of Ball ("The Concept of Dynamic Analysis," Bell Laboratories Lucent Technologies, 1999).

c) Whether claims 2 and 16 were properly rejected under 35 U.S.C. §103(a) as being unpatentable over Ur, Cahill, Benlarbi, Mitchell et al., and Kuzmin et al.

(7) Arguments

A. The rejection of claims 1, 4, 5, 8-12, 15, and 17 under 35 U.S.C. §103(a) as being unpatentable over Ur et al. in view of Cahill et al., Benlarbi et al., and Mitchell et al. should be reversed

The test for determining if a claim is rendered obvious by one or more references for purposes of a rejection under 35 U.S.C. § 103 is set forth in *KSR International Co. v. Teleflex Inc.*, 550 U.S._, 82 USPQ2d 1385 (2007):

> "Under §103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or nonobviousness of the subject matter is determined. Such secondary considerations as commercial success, long felt but unsolved needs, failure of others, etc., might be utilized to give light to the circumstances surrounding the origin of the subject matter sought to be patented." Quoting *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1 (1966).

As set forth in MPEP 2143.03, to ascertain the differences between the prior art and the claims at issue, "[a]ll claim limitations must be considered" because "all words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385. According to the Examination Guidelines for Determining Obviousness Under 35 U.S.C. 103 in view of *KSR International Co. v. Teleflex Inc.*, Federal Register, Vol. 72, No. 195, 57526, 57529 (October 10, 2007), once the aforementioned *Graham* factual inquiries are resolved, there must be a determination of whether the claimed invention would have been obvious to one of ordinary skill in the art based on any one of the following proper rationales:

> (A) Combining prior art elements according to known methods to yield predictable results; (B) Simple substitution of one known element for another to obtain predictable results; (C) Use of known technique to improve similar

> devices (methods, or products) in the same way; (D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (E) "Obvious to try"—choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; (F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art; (G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention. *KSR International Co. v. Teleflex Inc.*, 550 U.S._, 82 USPQ2d 1385 (2007).

Furthermore, as set forth in *KSR International Co. v. Teleflex Inc.*, quoting from *In re Kahn*, 441 F. 3d 977, 988 (CA Fed. 2006), "[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasonings with some rational underpinning to support the legal conclusion of obviousness."

Therefore, if the above-identified criteria and rationales are not met, then the cited reference(s) fails to render obvious the claimed invention and, thus, the claimed invention is distinguishable over the cited reference(s).

Independent Claims 1, 10, 15, and 17

It is respectfully submitted that the examiner failed to ascertain the differences between the prior art and at least the independent claims at issue because the prior art does not show those elements as the examiner alleged. Specifically, independent claims 1, 10, 15 and 17 all recite,

> determining a static use count of said system classes...;
> deriving a dynamic use count of each of said system classes...;
> assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count....

It is respectfully submitted that the specification clearly defines the terms "static use count" and "dynamic use count" of system classes in at least page 8, lines 9-31. According to MPEP 2111.01, part IV,

> An applicant is entitled to be his or her own lexicographer and may rebut the presumption that claim terms are to be given their ordinary and customary meaning by clearly setting forth a definition of the term that is different from its ordinary and customary meaning(s).
>
> ...
>
> Where an explicit definition is provided by the applicant for a term, that definition will control interpretation of the term as it is used in the claim.

The examiner admitted that Ur et al. does not disclose the aforementioned claimed features as clearly defined in the specification. Office Action, p. 3. However, the Office Action attempted to cure these defects by alleging that Cahill et al., Benlarbi et al., and Mitchell et al. together disclose such claimed features. Particularly, the examiner alleged that Cahill et al. discloses both static and dynamic use count of each of said system classes in Section 3.4 for Method Inheritance Factor (MIF) and Section 3.5 for Polymorphism Factor (PF), with the inheritance factor metric corresponding to the claimed static use count and the polymorphism factor metric corresponding to the claimed dynamic use count.

It is respectfully submitted that Cahill et al. is directed to a Java Metrics Reporter (JMR) used for software analysis to determine the complexity of a software. To that effect, the JMR employs a number of pre-selected metrics for measuring software complexity, such as the Basic metrics, complexity metrics, inheritance metrics, and polymorphism metrics, as stated in Section 3.1 and cited by the Office Action. However, Cahill et al. provides no discussion regarding a reliance on a *determined static use count of system classes* and *derived dynamic use count of each of the system classes* in order to assign a proportional weighing attribute to each system class. At best, Cahill et al. mentions the use of a complexity metric called "weighted methods per class (WMC)" in Section 3.3. However, that metric is for measuring the sum of complexities of all methods of a class, and not for determining or deriving the static and dynamic use counts of each of the system classes as claimed, nor for assigning a proportional weighing attribute to each system class based on such counts.

As for the MIF in Section 3.4 of Cahill et al. as cited in the Office Action, it is an inheritance factor metric that is defined as "the proportion of the inherited methods of a class relative to all available methods of that class, averaged over all classes in the system." *See* Cahill et al., Section 3.4. Thus, the MIF and inheritance factor metric have nothing to do with *determining a static use count of a class* as the examiner alleged, and much less to do with *deriving a dynamic use count of a class* as also claimed. As for the PF in Section 3.5 of Cahill et al., as cited in the Office Action, it is a polymorphism factor metric that is defined as the proportion of the number of polymorphic situations of a class..., relative to the maximum potential number of polymorphic situations occurs when every method in a class is overridden by every one of its descendents." *See* Cahill et al., Section 3.5. Thus, the PF has nothing to do with *deriving a dynamic use count of a class* as the examiner alleged, and much less to do with *determining a static use count of a class* as also claimed.

It is respectfully submitted that neither Benlarbi et al., Ball, nor Mitchell et al. cures the aforementioned defects found in Ur et al. and Cahill et al. Indeed, Benlarbi et al. discusses the use of polymorphism measures for early risk prediction that may be based on run-time binding (allegedly dynamic) decisions and compile time (allegedly static) linking decision. However, there is no mention of any determination or derivation of both static and dynamic use counts of the system classes and assignment of a proportional weighing attribute to the system classes based on such counts as claimed. Ball discusses the concept of only dynamic analysis with no mention of any determination or derivation of both static and dynamic use counts of system classes and assignment of a proportional weighing attribute to the system classes based on such counts. Likewise, Mitchell et al. teaches the use of coupling metrics CBO that is defined as the number of classes to which one of the classes is coupled. *See* Mitchell et al., page 4, first column. Thus, such a definition is not the same as a determination or derivation of both static and dynamic use counts of system classes.

In the Examiner's Response section on pages 25-26 of the Office Action, the examiner alleged that Mitchell et al., and not Cahill et al., was used to show that both static and dynamic metrics are used to properly evaluate the behavior of an application at run time, and such a showing somehow motivates one skilled in the art to leap to the use of *a static use count of the system classes* as claimed from a mere reading of the inheritance factor metric in Cahill et al. and the use of *a dynamic use count of the system classes during application operation* as claimed from a mere reading of the polymorphism factor metric in Cahill et al.

As noted above, the examiner alleged that Cahill et al. uses the inheritance factor metric as a static metric and the polymorphism factor metric as a dynamic metric. Thus, from the teaching of Mitchell et al., as also alleged by the examiner, one of ordinary skill in the art would have been motivated to use both the inheritance and polymorphism factor metrics together in Cahill et al. to properly evaluate the behavior of an application at run time. However, it is respectfully submitted that such a motivation to combine Cahill et al. with Mitchell et al. (along with the primary reference, Ur et al.) does not negate the fact that the *particular* static metric, namely, the claimed *static use count of the system classes* as claimed, and the *particular* dynamic metric, namely, the claimed *dynamic use count of the system classes during application operation* as claimed, remain elusive to the resulting combination.

Because the examiner failed to ascertain the actual differences between independent claims 1, 10, 15, 17 and the cited references, their combination would not have made obvious *all* features recited in these claims. Accordingly, it is respectfully submitted that the Office Action failed to establish a *prima facie* case of obviousness against these claims and their corresponding dependent claims, and the rejection of claims 1, 2, and 4-17 under 35 U.S.C. § 103 should be reversed.

B. The rejection of claims 6 and 13 under 35 U.S.C. §103(a) as being unpatentable over Ur et al. in view of Cahill et al., Benlarbi et al., Mitchell et al., and Ball should be reversed

For at least the above reasons, it is respectfully submitted that there are actual differences between the independent claims and the already-cited references that the examiner failed to consider. Thus, the examiner did not rely upon Ball to make up for the deficiencies in the already-cited references to address these actual differences. Indeed, the examiner relied on Ball to allegedly show the assignment of static and dynamic observation percentages to each system class - not to show the use of *static use count of the system classes* and *dynamic use count of the system classes during application operation* as claimed.

Accordingly, it is respectfully submitted that the rejection of claims 6 and 13 under 35 U.S.C. § 103 should be reversed as well.

C. The rejection of claims 2 and 16 under 35 U.S.C. §103(a) as being unpatentable over Ur et al. in view of Cahill et al., Benlarbi et al., Mitchell et al., and Kuzmin et al. should be reversed

For at least the above reasons, it is respectfully submitted that there are actual differences between the independent claims and the already-cited references that the examiner failed to consider. Thus, the examiner did not rely upon Kuzmin et al. to make up for the deficiencies in the already-cited references to address these actual differences. Indeed, the examiner relied on Kuzmin et al. to allegedly show the testing of the most heavily weighted portion of system classes - not to show the use of *static use count of the system classes* and *dynamic use count of the system classes during application operation* as claimed.

Accordingly, it is respectfully submitted that the rejection of claims 6 and 13 under 35 U.S.C. § 103 should be reversed as well.

(8) Conclusion

For at least the reasons given above, the rejections of claims 1, 2, and 4-17 are improper. Accordingly, it is respectfully requested that such rejections by the Examiner be reversed and these claims be allowed. Attached below for the Board's convenience is an Appendix of claims 1, 2, and 4-17 as currently pending.

Please grant any required extensions of time and charge any fees due in connection with this Appeal Brief to deposit account no. 08-2025.

Respectfully submitted,

Dated: April 10, 2008 By

Tiep H. Nguyen
Registration No.: 44,465

Ashok K. Mannava
Registration No.: 45,301

MANNAVA & KANG, P.C.
11240 Waples Mill Road
Suite 300
Fairfax, VA 22030
(703) 652-3819 (or 3822)
(703) 865-5150 (facsimile)

(9) Claim Appendix

1. A method for testing software, comprising:

examining an application software program including calls to system classes with both a static analysis tool and a dynamic analysis tool;

determining a static use count of said system classes from the examining;

deriving a dynamic use count of each of said system classes during operation of said application software program from the examining;

assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count; and

testing said system classes in order according to said corresponding proportional weighing attributes.

2. The method of claim 1, wherein:

the step of testing is such that only the most heavily weighted portion of all such system classes are tested at all.

4. The method of claim 1, wherein:

producing a static use count further comprises assigning a static observation percentage to each system class by dividing said static use count by a sum of all static use counts.

5. The method of claim 1, wherein:

producing a dynamic use count further comprises assigning a dynamic observation percentage to each system class by dividing said dynamic use count by a sum of all dynamic use counts.

6. The method of claim 1, wherein:

producing a static use count further comprises assigning a static observation percentage to each system class by dividing the static use count by a sum of all static use counts; and

producing a dynamic use count further comprises assigning a dynamic observation percentage to each system class by dividing the dynamic use count by a sum of all dynamic use counts.

7. The method of claim 6, wherein the step of assigning to each of the system classes a weight based on the static use count and the dynamic use count further comprises the steps of:

assigning to a public untested system class in the system classes a first weight defined by a first constant plus a sum of the static use count plus the dynamic use count;

assigning a private untested software class in the system classes a second weight that is equal to the first constant;

assigning to each public function in the system classes that is not fully tested a third weight that is defined as a second constant that is less than the first constant, to which is added a sum of the static observation percentage plus the dynamic observation percentage; and assigning to all remaining public and private functions in the system classes a fourth weight defined as a third constant that is less than the second constant.

8. The method of claim 1, wherein:

the testing the system classes further comprises ending a test when a testing resource is exhausted and prior to testing a last entry having a least weight.

9. The method of claim 8, wherein:

the testing the system classes further comprises ending a test when at least a limit of available time or funding is exhausted and prior to testing a last entry having a least weight.

10. A machine-readable medium on which is encoded machine-readable code for testing object-oriented system software having system classes, the machine readable code comprising:

machine-readable code for running a static analysis tool for examining an application software program, the application software program including calls to the system classes;

machine-readable code for determining a static use count of the system classes in the application software program from the result;

machine-readable code for running a dynamic analysis tool for examining the application software program and producing a dynamic use count based on the application software program's dynamic use of the system functions while running the application software program;

machine-readable code for assigning to each system class a weight based on the static use count and the dynamic use count, and

machine-readable code for testing the system classes, in order, based on the assigned weight, from a first entry having a greatest weight.

11. The software of claim 10, wherein:

producing a static use count further comprises assigning a static observation percentage to each system class by dividing the static use count by a sum of all static use counts.

12. The software of claim 10, wherein:

producing a dynamic use count further comprises assigning a dynamic observation percentage to each system class by dividing the dynamic use count by a sum of all dynamic use counts.

13. The software of claim 10, wherein:

producing a static use count further comprises assigning a static observation percentage to each system class by dividing the static use count by a sum of all static use counts, and

producing a dynamic use count further comprises assigning a dynamic observation percentage to each system class by dividing the dynamic use count by a sum of all dynamic use counts.

14. The software of claim 10, wherein the assigning to each of the system classes a weight based on the static use count and the dynamic use count further comprises the steps of:

assigning to a public untested system class in the system classes a first weight defined by a first constant plus a sum of the static use count plus the dynamic use count;

assigning a private untested software class in the system classes a second weight that is equal to the first constant;

assigning to each public function in the system classes that is not fully tested a third weight that is defines as a second constant that is less than the first constant, to which is added a sum of the static observation percentage plus the dynamic observation percentage; and

assigning to all remaining public and private functions in the system classes a fourth weight defined as a third constant that is less than the second constant.

15. A machine-readable medium on which is encoded a software tester program code, the software tester program code comprising:

means for examining an application software program including calls to system classes with both a static analysis tool and a dynamic analysis tool;

means for determining a static use count of said system classes from the examining;

means for deriving a dynamic use count of each of said system classes during operation of said application software program from the examining;

means for assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count; and means for testing said system classes in order according to said corresponding proportional weighing attributes.

16. The tester of claim 15, wherein:

the means for testing is such that only the most heavily weighted portion of all such system classes are tested at all.

17. A business model for testing software, comprising:

setting a resource limit on the available time or money that is devoted to testing a particular application software program;

examining said application software program including calls to system classes with both a static analysis tool and a dynamic analysis tool;

determining a static use count of said system classes from the examining;

deriving a dynamic use count of each of said system classes during operation of said application software program from the examining;

assigning a proportional weighing attribute to each system class based on its corresponding static use count and dynamic use count;

testing said system classes in order according to said corresponding proportional weighing attributes and proceeding down to the least heavily weighted system classes; and

stopping testing when said resource limit is reached.

(10) Evidence Appendix

None.

(11) Related Proceedings Appendix

Nonc.